PATENT APPLICATION
Docket No. 13768.444

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of		)
	Cliff M. R. Don et al.	)
Serial No.:	10/602,126	) Art Unit) 2168
Filed:	June 24, 2003	)
Conf. No.:	1587	)
For:	DATABASE DRIVEN TYPE EXTENSIBILITY	)
Examiner:	Jay A. Morrison	)
Customer No.:	047973	)

AMENDMENT "B"

Mail Stop AMENDMENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In response to the Office Action of June 29, 2006, please amend the above-identified application as follows:

Amendments to the Specification begin on page 2 of this paper.

Amendments to the Abstract begin on page 3 of this paper.

Amendments to the Claims begin on page 4 of this paper.

Remarks/Arguments begin on page 17 of this paper.

AMENDMENTS TO THE SPECIFICATION

In paragraph [026] of the originally-filed application, please amend as reflected in the following marked-up version of the paragraph:

[0026] As should be appreciated from the ~~forgoing~~foregoing, the code and definitions associated with the new data types can be modified and obtained from a single and centralized location, such that each of the middle tier servers does not have to continually monitor each of the other middle tier servers for new data types and corresponding updates. Accordingly, new data types and corresponding updates no longer have to be manually ~~depoloyed~~deployed to each of the middle tier servers within the distributed multi-tier system. Conformity between the various middle tier servers can also be realized. In particular, since the code and definitions of the data types are obtained from a single location, the risk of having multiple different and inconsistent definitions and data types is reduced. Additional advantages and features of the invention will be described in more detail below and can be realized by practicing the invention.

In paragraph [039] of the originally-filed application, please amend as reflected in the following marked-up version of the paragraph:

[0039] The next illustrated act, act 230, includes the creation or modification of logic modules in the one or more middle tier servers that are linked to the back end server so that they can utilize the extended assemblies provided by the back end server~~, act 230~~. The logic modules described herein, and illustrated in FIG. 1 as modules 150 and 160, generally include computer-executable instructions for enabling the middle tier servers to communicate with the back end server in push and/or pull environments, and to receive the extended assemblies that define and enable data types to be utilized by the middle tier servers.

AMENDMENTS TO THE ABSTRACT

In the Abstract of the originally-filed application, please amend as reflected in the following marked-up version of the paragraph:

Data types can be created, modified and deployed in multi-tier database systems by using extended assemblies. The extended assemblies are created by a back end server using the data and code contained in special tables and object tables of the back end server. The back end server determines which data types are to be deployed in the system, based on push or pull conditions. Thereafter, the corresponding extended assemblies are to be obtained and sent to one or more middle tier servers that utilize the extended assemblies to use the data types. However, some of the middle tier servers may need to be configured with appropriate logic modules ~~to~~ prior to utilizing the extended assemblies. A data type can be created or modified by creating or editing data and code contained in the special table and object tables prior to creating the extended assemblies.

AMENDMENTS TO THE CLAIMS

This listing of claims replaces all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) In a multi-tier server system that includes a back end server at a first tier and one or more additional servers at a middle tier, each additional server using multiple types of data objects that must be defined on the one or more additional servers before the data objects can be used by the one or more middle tier servers, a method for deploying one or more data types from the back end server to the one or more middle tier servers in a manner that maintains consistency and compatibility in the definitions of the data types and in code associated with each data type as stored on each middle tier server in the system, the method comprising:

an act of creating a special table in a database of the back end server, the special table including one or more fields for storing data identifying data types and corresponding code for enabling use of each of the data types, and the back end server acting as a single and centralized source from which all middle tier servers obtain data types and the corresponding code required to enable use of the data types by the one or more middle tier servers;

an act of identifying a data type to be deployed from the back end server to the one or more middle tier servers;

an act of obtaining an extended assembly that corresponds to the data type to be deployed, the extended assembly including data obtained from the special table, including data identifying the data type, one or more definitions of the data type, and the code for enabling ~~use~~ processing of ~~the data type~~data corresponding to the data type; and

an act of transmitting the extended assembly to the one or more middle tier servers in the multi-tier system such that the data type, as transmitted to and received by the one or more middle tier servers in the multi tier system, is consistent and compatible with a data type of the same kind stored on other middle tier servers in the system.

2. (Currently Amended) A method as recited in claim 1, further including an act of creating logic modules in the one or more middle tier servers that enable ~~utilization~~ the one or more middle tier servers to query for~~of~~ the extended ~~as~~ assembly.

3. (Original) A method as recited in claim 1, wherein the back end server includes a relational database.

4. (Previously Presented) A method as recited in claim 3, wherein the back end server comprises an SQL server.

5. (Original) A method as recited in claim 1, wherein the one or more middle tier servers includes an email server.

6. (Cancelled).

7. (Currently Amended) A method as recited in claim 1, wherein the act of identifying the data type to be deployed includes determining that the one or more middle tier servers has requested ~~or~~ the extended assembly, since the one or more middle tier servers ~~does~~ are not yet enabled for ~~use of~~ the data type.

8. (Currently Amended) A method as recited in claim 7, further including an act of adding a new middle tier server to the multi-tier system, and wherein the new middle tier server comprises the one or more middle tier servers that has requested the extended assembly~~or does not yet enable use of the data type~~.

9. (Original) A method as recited in claim 1, further including an act of creating one or more object tables that are linked to the special table and that include additional information defining the data type to be deployed, such that the extended assembly also includes the additional information.

10. (Currently Amended) In a multi-tier server system that includes a back end server at a first tier and one or more additional servers at a middle tier, each additional server using multiple types of data objects that must be defined on the one or more additional servers before the data objects can be used by the one or more middle tier servers, a method for deploying one or more data types from the back end server to the one or more middle tier servers in a manner that maintains consistency and compatibility in the definitions of the data types and in code associated with each data type as stored on each middle tier server in the system, the method comprising:

an act of modifying a special table in a database of the back end server, the special table including one or more fields for storing data that identifies data types and includes corresponding code for enabling use of each of the data types, and the back end server acting as a single and centralized source from which all middle tier servers obtain data types and the corresponding code required to enable use of the data types by the one or more middle tier servers, the act of modifying including at least one of modifying the stored data within the one or more fields and adding new stored data to the one or more fields;

an act of identifying a data type to be deployed from the back end server to the one or more middle tier servers;

an act of obtaining an extended assembly that corresponds to the data type to be deployed, the extended assembly including at least one of the modified stored data and the new stored data as obtained from the special table, including data identifying the data type, and the executable code that, when executed, ~~for~~ enabl~~ing~~es ~~use~~ the one or more middle tier servers to process~~of~~ the modified stored data or the new stored data associated with the data type; and

an act of transmitting the extended assembly to the one or more middle tier servers in the multi-tier system such that the data type as transmitted to and received by the one or more middle tier servers in the multi tier system is consistent and compatible with a data type of the same kind stored on other middle tier servers in the system.

11. (Currently Amended) A method as recited in claim 10, further including an act of determining which of the one or more middle tier servers should be sent the extended assembly.

12. (Currently Amended) A method as recited in claim 11, wherein determining which of the one or more middle tier servers should be sent the extended assembly ~~the extended assembly enables use of the data type to be deployed at the one or more middle tier servers that have been determined to be sent the extended assembly~~comprises the acts of:

sending data associated with the data type to the one or more middle tier servers; and

receiving one or more requests for the extended assembly from the one or more middle tier servers upon the one or more middle tier servers identifying that the data associated with the data type cannot be processed at the one or more middle tier servers.

13. (Original) A method as recited in claim 10, wherein the back end server includes a relational database.

14. (Previously Presented) A method as recited in claim 10, wherein the back end server comprises an SQL server.

15. (Original) A method as recited in claim 10, wherein the one or more middle tier servers includes an email server.

16. (Original) A method as recited in claim 10, wherein the act of modifying includes adding new stored data corresponding to a new data type not previously enabled in the multi-tier system prior to adding the new stored data.

17. (Currently Amended) In a multi-tier server system that includes a back end server at a first tier and one or more additional servers at a middle tear, each additional server using multiple types of data objects that must be defined on the one or more additional servers before the data objects can be used by the one or more middle tier servers, a method for deploying one or more data types from the back end server to the one or more middle tier servers in a manner that maintains consistency and compatibility in the definitions of the data types and in code associated with each data type as stored on each middle tier server in the system, the method comprising:

an act of adding a new middle tier server to the multi-tier system, the new middle tier server being configured to utilize extended assemblies that are obtained from the back end server which acts as a single and centralized source from which all middle tier servers obtain data types and corresponding code required to enable use of the data types by the one or more middle tier servers, the extended assemblies being configured to enable the use of one or more data types that are defined by data and enabled by executable code that is contained in the extended assemblies;

an act of determining which of the one or more data types are to be deployed from the back end server to the new middle tier server, wherein the act of determining is based at least in part on a request by the new middle tier server for data to enable use of one or more data types;

an act of obtaining one or more extended assemblies corresponding to the one or more data types that have been determined to be deployed, each of the one or more extended assemblies including data and executable code obtained from a special table stored in a database of the back end server, the special table including one or more fields for storing data identifying data types and corresponding code for ~~enabling use of each~~processing data associated with ~~of~~ the data types; and

an act of transmitting, to the new middle tier server, the one or more extended assemblies that correspond to the one or more data types that have been determined to be deployed, such that the one or more data types as transmitted to, and received by, the new middle tier server are consistent and compatible with one or more data types of the same kind on other middle tier servers in the system, and which were received by the other middle tier servers from the back end server.

18. (Currently Amended) A method as recited in claim 17, wherein the act of determining is further based at least in part on the ~~capabilities of the~~ new middle tier server identifying what other data types are supported, and identifying that the one or more data types to be deployed are not supported at the new middle tier server.

19. (Canceled) A method as recited in claim 17, wherein the act of determining is based at least in part on a request by the new middle tier servers for data to enable use of one or more data types.

20. (Cancelled).

21. (Currently Amended) In a multi-tier server system that includes a back end server at a first tier and one or more additional servers at a middle tier, each additional server using multiple types of data objects that must be defined on the one or more additional servers before the data objects can be used by the one or more middle tier servers, a method for deploying one or more data types from the back end server to the one or more middle tier servers in a manner that maintains consistency and compatibility in the definitions of the data types and in code associated with each data type as stored on each middle tier server in the system, the method comprising:

an act of creating a special table in a database of the back end server, the special table including one or more fields for storing data identifying a data type and corresponding executable code for enabling ~~use~~ processing of data associated with~~of~~ the data type, and the back end server acting as a single and centralized source from which all middle tier servers obtain data types and the corresponding code required to enable use of the data types by the one or more middle tier servers;

a step for deploying the data type from the back end server to the one or more middle tier servers, upon request, such that the data type as transmitted to and received by the one or more middle tier servers in the multi-tier server system is consistent and compatible with a data type of the same kind stored on other middle tier servers in the system.

22. (Currently Amended) A method as recited in claim 21, wherein the step for deploying the data type to the one or more middle tier servers upon request comprises corresponding acts that include:

an act of identifying the data type to be deployed based on receipt of the data type at the one or more middle-tier servers, and the one or more middle-tier servers requesting an extended assembly for the data type since the data type cannot be processed at the one or more middle tier servers;

an act of obtaining an extended assembly that corresponds to the data type to be deployed, the extended assembly including the data from the special table identifying the data type and the executable code for enabling processing of the data associated with~~use of~~ the data type; and

an act of transmitting the extended assembly to the one or more middle tier servers in the multi-tier system that requested the extended assembly.

23. (Original) A method as recited in claim 22, further including an act of creating logic in the one or more middle tier servers that enables utilization of the extended assembly.

24. (Original) A method as recited in claim 22, further including an act of creating at least one object table that includes at least some information defining the data type, and wherein the extended assembly includes the at least some information.

25. (Currently Amended) A computer program product for use in a multi-tier server system that includes a back end server at a first tier and one or more additional servers at a middle tier, each additional server using multiple types of data objects that must be defined on the one or more additional servers before the data objects can be used by the one or more middle tier servers, the computer program product including one or more computer-readable media having computer-executable instructions for implementing a method for deploying one or more data types from the back end server to the one or more middle tier servers in a manner that maintains consistency and compatibility in the definitions of the data types and in code associated with each data type as stored on each middle tier server in the system, the method comprising:

an act of creating a special table in a database of the back end server, the special table including one or more fields for storing data identifying data types and corresponding code for enabling use of each of the data types, and the back end server acting as a single and centralized source from which all middle tier servers obtain data types and the corresponding code required to enable use of the data types by the one or more middle tier servers;

an act of identifying a data type to be deployed from the back end server to one or more middle tier servers;

an act of obtaining an extended assembly that corresponds to the data type to be deployed, the extended assembly including data obtained from the special table, including data identifying the data type, one or more definitions of the data type, and the code for enabling ~~use~~ processing of data associated with the data type; and

an act of transmitting the extended assembly to the one or more middle tier servers in the multi-tier system such that the data type as transmitted to and received by the one or more middle tier servers in the multi tier system is consistent and compatible with a data type of the same kind stored on other middle tier servers in the system.

26. (Currently Amended) A computer program product as recited in claim 25, wherein the method further includes an act of creating logic modules in the one or more middle tier servers that enable ~~utilization~~ the one or more middle tier servers to query for~~of~~ the extended assembly.

27. (Previously Presented) A computer program product as recited in claim 25, wherein the back end server includes an SQL server.

28. (Original) A computer program product as recited in claim 25, wherein the one or more middle tier servers includes an email server.

29. (Cancelled).

30. (Currently Amended) A computer program product as recited in claim 25, wherein the act of identifying the data type to be deployed includes determining that the one or more middle tier servers has requested ~~or~~ the extended assembly, since the one or more middle tier servers ~~does~~ are not yet enabled ~~use of~~for the data type.

31. (Currently Amended) A computer program product as recited in claim 25, wherein the method further includes an act of adding a new middle tier server to the multi-tier system, and wherein the new middle tier server comprises the one or more middle tier servers that has requested ~~or does not yet enable use of the data type~~the extended assembly.

32. (Original) A computer program product as recited in claim 25, wherein the method further includes an act of creating one or more object tables that are linked to the special table and that include additional information defining the data type to be deployed, and wherein the extended assembly also includes the additional information.

33. (Currently Amended) A computer program product as recited in claim 32, wherein the method further includes modifying at least one of the special table and the one or more object tables.

34. (Previously Presented) A method as recited in claim 1, wherein the extended assembly is a single data structure that includes all the data required to enable the one or more middle tier servers to use the data type.

35. (New) A method as recited in claim 1, wherein the one or more middle tier servers have limited program code means to process data associated with less than all of the data types in the multi-tier system, and the back end server has all program code means to process any data associated with all of the data types in the multi-tier system.

36. (New) A method as recited in claim 35, wherein the one or more middle tier servers are only equipped to recognize and process data objects associated with a particular data type when program code means comprising executable machine code of the extended assembly for the particular data type has been received from the back end server and installed at the one or more middle tier servers.

37. (New) At a middle tier server in a multi-tier database server system that includes a back end database server at a first tier and one or more additional database servers at a middle tier, wherein the middle tier server is configured to process data corresponding to data types defined by the back end server at the first tier, a method for deploying one or more data types from the back end server at the middle tier server in a manner that maintains consistency and compatibility in the definitions of the data types and in code associated with each data type in the multi-tier database server system, the method comprising:

an act of receiving at a middle tier server one or more data objects from a back end server, the one or more received data objects associated with at least one data type;

an act of initiating one or more processing functions for the one or more received data objects associated with the at least one data type;

an act of identifying that the at least one data type of the one or more data objects is not recognized, such that the initiated one or more initiated processing functions have failed at the middle tier server;

an act of pulling one or more extended assemblies corresponding to the at least one data type from the back end server; and

an act of processing the one or more data objects associated with the at least one data type using the pulled one or more extended assemblies, wherein the middle tier server successfully recognizes the at least one data type, and successfully processes the one or more received data objects associated with the at least one data type.

38. (New) A method as recited in claim 37, wherein the one or more pulled extended assemblies comprise computer-executable instructions that, when executed at the middle tier server, cause one or more processors at the middle tier server to format the one or more data objects so that the one or more data objects can be processed.

REMARKS

With this paper, independent method claims 1, 10, 17 and 21, and corresponding computer program product claim 25 have been amended, together with various dependent claims (2, 7-8, 11-12, 18, 22, 25-26, 30, 31, 33), while claims 6, 19, and 29 have been cancelled and claims 35-38 have been added. Accordingly, claims 1-5, 7-18, 21-28, and 30-38 are presented for reconsideration, of which claims 1, 10, 17, 21, 25, and 37 are independent.

The most recent Office Action ("*Office Action*") objected to dependent claims 5 and 6, as well as claims 28 and 29, as being substantial duplicates of each other. As is reflected in the above claim listing, the objected-to claims 6 and 29 have each been cancelled, and, as such, the object of record is now moot.

In addition, the *Office Action* rejected claims 1-3, 7-13. 16-29, 21-26, and 30-34 under 35 U.S.C. § 103(a), as being unpatentable over U.S. Patent Publication No. 2002/0188610 ("*Spencer*") in view of U.S. Patent No. 6,889,229 ("*Wong*").[1] The *Office Action* also rejected various dependent claims under 35 U.S.C. § 103(a) as obvious over Wong in combination with U.S. Patent No. 6,578,068 ("*Bowman-Amuah*").

In general, *Wong* describes a method for "peer-to-peer replication of objects" between various nodes connected over a network. To this end, *Wong* discloses "mapping" various object attributes from one sending node to a receiving node, so that unique data fields (and corresponding data) at the sending node can be correlated to data fields at the receiving node. If properly mapped and correlated, the receiving node can then receive (*i.e.*, copy over) and process data from the sending node. Otherwise, if not properly mapped, the data from the receiving node may not be properly copied at all, and, at the very least, cannot be processed. *Compare* col. 3, ll. 10-15, *with* col. 4, l. 49 – col. 5, l. 4; col. 9, ll. 25-30.

According to *Wong*, therefore, the sending node creates a replication group of objects, including any user-defined objects. Col. 6, ll. 41-56; Col. 8, ll. 25-43. Thereafter, the sending node copies data defining the user-defined object(s) to a data structure on the second node. Col. 7, ll. 58-65; col. 9, ll. 25-31. This data generally includes naming correlation/mapping

[1] Since Wong qualifies as "prior" art, if at all, under 35 U.S.C. 102(e), applicants reserve the right to challenge the status of that reference as qualifying "prior" art. Accordingly, any statement or comment herein to Wong is made merely for purposes of argument, and assumes *arguendo* that the reference is proper qualifying prior art.

information, such as column designations for the user-defined type, names for the user-defined type, attribute correlations between the sending and receiving node(s), and so forth. Col. 8, ll. 55-65; col. 15, ll. 3-36. Only after sending this naming correlation (or "mapping") information to the receiving node can the sending node copy the user-defined object to the receiving node without failure, since the receiving node would then know where to place and/or otherwise organize the received data (*e.g.*, the receiving node can correlate a column name in which to place the user-defined data.) (Col. 7, ll. 58-65; col. 9, ll. 25-31; col. 9, ll. 46-51; col. 11, ll. 29-41; col. 19, ll. 51-65)

Importantly, however, the *Wong* reference discloses nothing about the receiving node's ability to *process* data associated with a particular data type, such as data specifically-formatted to be opened only by a particular type of application program. In particular, the sole reason the receiving node in the *Wong* reference appears unable to process certain data appears to be that the receiving node does not know where to place (*i.e.*, "fill" data fields with received data) certain unique (user-defined) data it receives from the sending node. Col. 3, ll. 10-15; col. 7, ll. 48-51; col. 9, ll. 45-51.

This contrasts in at least one respect with Applicant's claimed methods, which require a middle tier server to receive an "extended assembly" from a back end server. For example, upon encountering an unfamiliar data type, a middle tier server can receive program code means or computer-executable code, such as an application program or module, which allows the middle tier server to recognize the data and then process it. At the outset, therefore, and in contrast with the *Wong* reference, the middle tier server can actually receive any data successfully from another source at any time, even if the receiving middle tier server does not recognize the data.

As such, the *Wong* reference fails to disclose, teach, or otherwise suggest, whether singly or in combination with *Spencer* or *Bowman-Amuah*, an extended assembly that includes "data obtained from the special table, including data identifying the data type, one or more definitions of the data type, and the code for enabling *processing* of data corresponding to the data type," as recited in amended claim 1. In addition, the *Wong* reference fails to disclose, teach, or otherwise suggest, whether singly or in combination with *Spencer* or *Bowman-Amuah*, that the extended assembly includes "*executable code* that, when executed, enables the one or more middle tier servers to *process* the modified stored data or the new stored data associated with the data type," as recited in amended claim 10. Similar amendments are found in independent claims 17, 21, and 25.

As previously mentioned, the *Wong* reference teaches that data having user-defined types cannot even be copied to a receiving node unless the user-defined types (and/or corresponding attribute data) are correlated or mapped in some way with data fields at the receiving node. Thus, the *Wong* reference necessarily fails to disclose, teach, or otherwise suggest, whether singly or in combination with *Spencer* or *Bowman-Amuah* (and even teaches away from), determinations at the back end server "based at least in part on a request by the new middle tier server for data to enable use of one or more data types," as recited in amended claim 17. Similar limitations are found in amended dependent claims 2, 7, 12, 18, 22, 26, and 30.

Along these lines, Applicants have added new claims 35-38, which include new independent claim 37, to clarify Applicants' invention. Claims 37-38 are drafted particularly from the perspective of what steps are performed at or by a given middle tier server that encounters a data type it cannot recognize.

As a final matter, the *Office Action* cited column 14, ll. 23-34 for the limitation recited in claim 34, which generally recites that the extended assembly is a single data structure that includes all of the data required to use a particular type. Applicants respectfully traverse this characterization of the *Wong* reference, particularly as an "extended assembly" should be properly construed. For example, Applicant's disclosure teaches that an extended assembly is an application program or module that is used by a middle tier server to "process" data associated with a data "type," such as "types" of image data formats, word processing data formats, etc. *Compare* ¶¶ 23-24, *with* ¶ 43. In general, the term "assembly" is understood in the art as computer code that can be (or is) assembled or compiled into computer-understandable information for directing specified computer processes. Correlated to Applicant's invention, this means that if a middle tier server encounters digital image data for which it does not have the particular assembly (*e.g.*, application program or module) to process/open it, regardless of whether the digital image data is placed or filled in an appropriate database data structure, the middle tier server cannot process the digital image data.

By contrast, a "type," as understood from the *Wong* reference, appears to mean a kind of end-user data category that is meant to be interpreted or modified via end-user input/output, rather than, for example, as computer-readable data that is used by a computer-system to process data objects. For example, "Table 1" of the *Wong* reference shows that a "user-defined Type" includes the category of "street_address." As previously discussed, therefore, the information received from the sending computer simply maps various end-user-defined *categories*, and thus does not constitute a program or module used to direct a computer's processing of specific types/formats of computer-readable data objects (*e.g.*, new claim 38).

Even assuming, *arguendo*, that the extended assembly of Applicant's invention was analogous in some way to the information received from the sending node in the *Wong* reference, col. 14, ll. 23-34 does not teach a "single data structure" that can be used for all data associated with a data type. In particular, col. 14, ll. 23-34 discusses that a "single auxiliary leaf" (*i.e.*, for a "top-level" type) can be used as some sort of an index to indicate "the presence or absence" of "all contained user-defined types." Col. 14, ll. 25-27. As such, this single leaf does not contain the actual data (e.g., column number, column names, column field data) of other leafs, and can only be used in the context of other leafs received from the sending node. Applicants find no other teaching in the *Wong* reference to contradict this construction. Applicants, therefore, respectfully traverse the rejection of claim 34.

Thus, for at least the foregoing reasons, the rejection(s)/objection(s) of record are now moot.

Applicants therefore respectfully request favorable reconsideration and allowance of the pending claims. In the event the Examiner finds any remaining impediment to allowance that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney.

Dated this ____ day of ________________, 2006.

Respectfully submitted,

MICHAEL J. FRODSHAM
Registration No. 48,699
Attorney for Applicant
Customer No. 047973

MJF
AAM0000000410V001